Lec: 10

We studied in the last lecture 3 types of search algorithms
This 3 types search for the solution without having any information about the goal Location.
In This lecture we will study informed search in which we need to know information about the goal location (How far the goal location from the node I expand now) to know if I work on the correct way OR not.

* Informed Search

Here we will study 3 things (1 function , 2 Search Algorithms)

- Heuristics.
  Function takes state of state space (world), and gives a number represent for how far the goal location from my location according to doing a process.
- Greedy Search.
  Search algorithm use the idea of Heuristics
  During studying it, we will know that it is not optimal search algorithm.
- A* Search
  Search algorithm Collects all Last search algorithms Ideas to get very good search algorithm

* Graph Search.

Back to study Graph Search.

* Recap : Search

1. Search Problem.
   - States (Configurations of the world)
   - Actions and costs
   - Successor Function which says how the states respond to actions (world dynamics)
   - Start state & goal state.

2. Search Tree.
   - Nodes : represent plans for reaching states
   - Plans have Costs (Sum of action Costs)

3. Search Algorithm :
   - Systematically builds a search tree.
   - Chooses an ordering of the fringe (how we decided which partial plan to expand next)
   - Optimal : Finds least-Cost plans.

Example: Pancake Problem

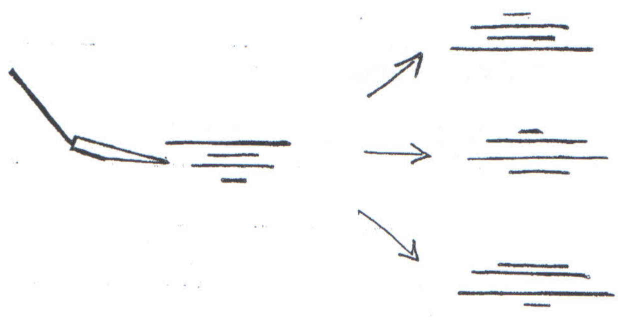
problem : need to arrange Pancakes From big to small

States : Shapes of pancakes during Flipping to reach to goal

Costs : Number of pancakes Flipped

Start State :

goal State :



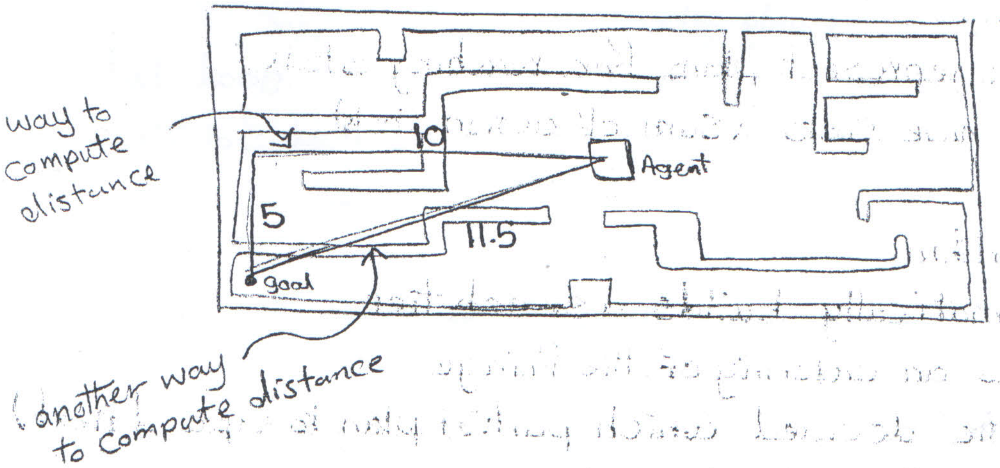
Statest

Algorithm : we can use UCS algorithm according to Cost
we also can use DFS or BFS

* Search Heuristics

- A heuristic is:
  - A function that estimates how close a state is to a goal
  - Designed for a particular Search problem.
  - Examples: Manhattan distance, Euclidean distance for pathing

* we make this process every step to know if I close to goal or not or if I walk on the right way or not.



Heuristic for this Example: distance between current state of Agent and goal

* every search problem need different heuristic (function) According to the nutural of this problem to know how close we are to the goal.

* We can get infinite number of heuristics to different infinite search problems.

* The Example of Pancake Problem
Heuristic: the number of the largest pancake that is still out of place.

5 Greedy Search.

Strategy: expand the node that seems closest to goal according to
- Heuristic : estimate of distance to nearest goal for each State.

Implementation: Fring is a priority queue (priority: lest heuristic)
Solution: lest heuristic solution

* This Algorithm doesn't care about Cost
* This Algorithm Cares only about heuristic
  So, In more cases we reach to lest heuristic goal but the very high Cost one so it isn't the optimal Solution.

- A Common Case:
  - Best-first takes you straight to the wrong goal.

- Worst-Case: like a badly-guided DFS

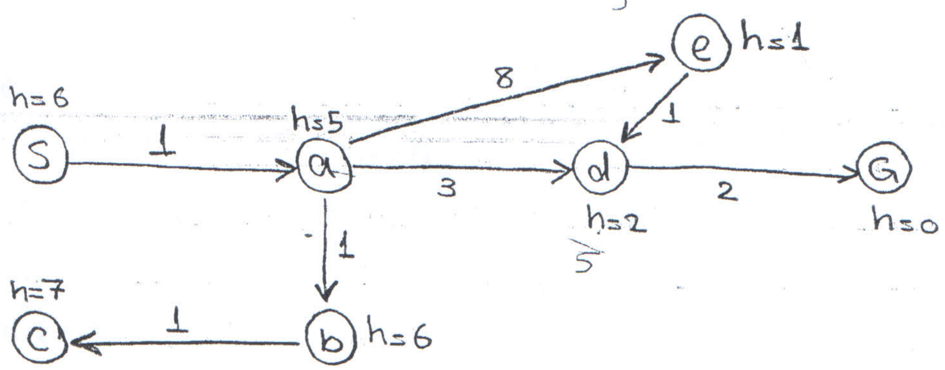
Solution Combining UCS and Greedy Search.

6 A* Search

Combining UCS and Gre dy Search

- Uniform-Cost orders by path Cost or backword cost g(n)
- Greedy orders by goal proximity, or Forward cost h(n)

Example:



Note : the value of h reduce when we close to goal

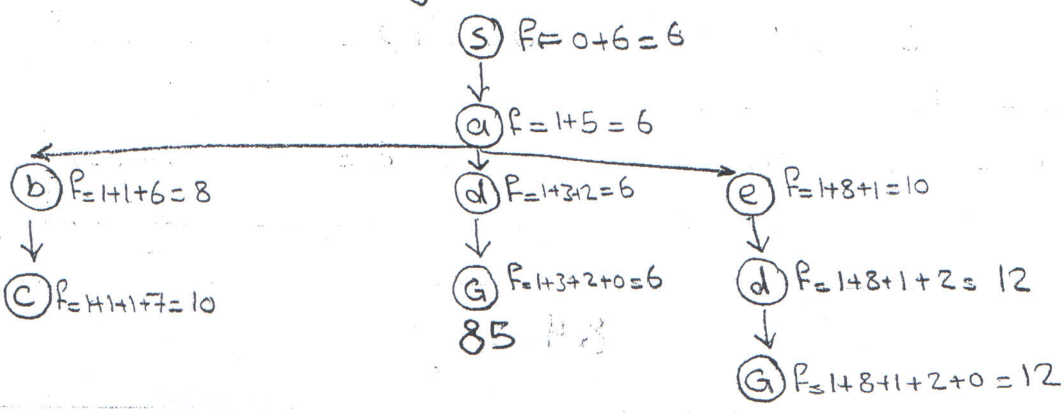
- Uniform-cost g(n)

  expand S → expand a → expand b → expand C (no sol
  backword to a → expand d → expand G
  ∴ Fringe (solution) S a d G

- Greedy h(n)

  expand S → expand e (h=1) → expand d → expand G
  ∴ fringe (solution) S e d G

- A* Search orders by the sum : $f(n) = g(n) + h(n)$



• A* search f(n)

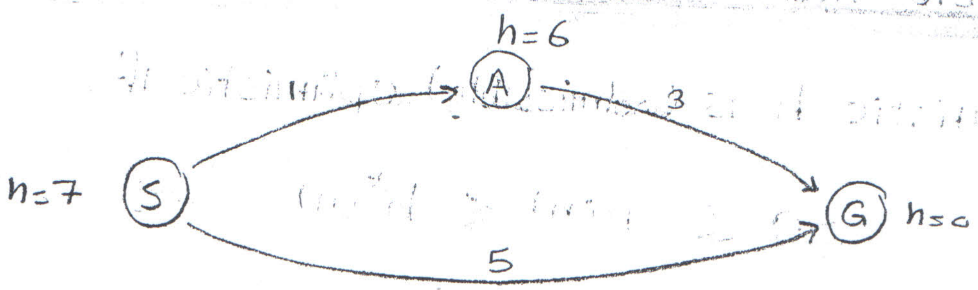
expand s ⟶ expand a ⟶ expand d ⟶ expand G

fringe s a d G

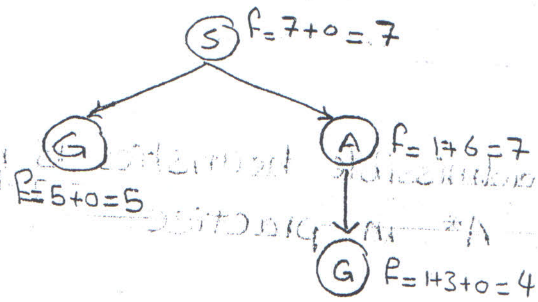
*• Should we stop when we enqueue a goal?

→ No; only stop when we dequeue a goal

* IS A* Optimal?



Solution



→ expand S ⟶ expand G (less f value)
we take the way of high cost

→ the value of h at A cause the problem (make me go to worst way)

→ The value of h should be logic to the value of cost

• what went wrong?

• Actual bad goal cost < estimated good goal cost

• we went estimates to be less than actual costs.

→ the value of h should be less or equal to cost to nearest goal from this path $0 < h < 4$

* Admissibility

  * Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe

  * Admissible (optimistic) heuristics slow down bad plans but plans but never outweigh true costs.

* Admissible Heuristics

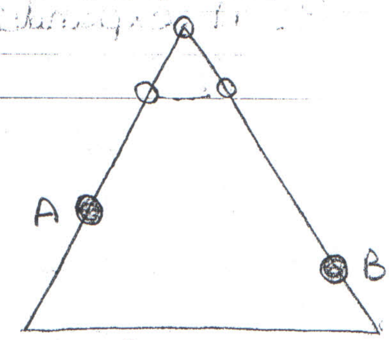
  • A heuristic h is (admissible) optimistic if:

$$0 \leq h(n) \leq h^*(n) \quad \Rightarrow h^*(n) = g(n)$$

where $h^*(n)$ is the true cost to a nearest goal
if h value is more than cost to nearest goal we can't get the optimal solution

  • Coming up with admissible heuristics is most of what's involved in using A* in practice.

* Optimality of A* Tree Search.



Assume

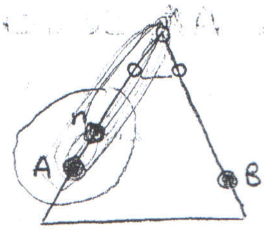
- A is an optimal goal node
- B is a sub optimal goal node
- h is admissible.

Claim:

. A will exit the fringe before B

Proof:

- Imagine B is on the fringe
- Some ancestor n of A is on the fringe, too (maybe A!)



- Claim: n will be expanded before B
  1. f(n) is less or equal to f(A)

Note: Deffinition of f-cost Admissibility of h

$h = 0$ at goal.

$\rightarrow h(n) \leq h^*(n) \quad \Rightarrow \underbrace{h^*(n)}_{\text{cost}} = g(A) - g(n)$

$h(n) \leq g(A) - g(n)$

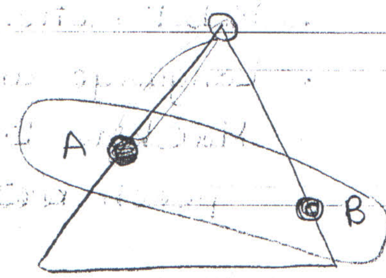
$\therefore h(n) + g(n) \leq g(A) \quad \Rightarrow f(n) = h(n) + g(n)$

$\therefore f(n) \leq g(A)$

$\rightarrow \because g(A) = f(A)$ where $h(\text{goal}) = 0$

$$\boxed{\therefore f(n) \leq f(A)}$$

2. f(A) is less than f(B)



$\because g(A) < g(B)$

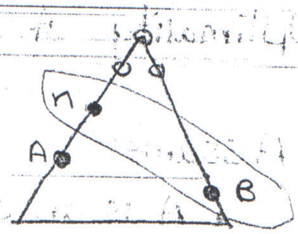
where B is suboptimal

$\because$ A and B are goals $\therefore h(A) = h(B) = 0$

$\therefore f(A) = h(A) + g(A) = g(A)$ & $f(B) = g(B)$

$$\boxed{\therefore f(A) < f(B)}$$

3. n expands before B



from 1 $f(n) \leq f(A)$

$\therefore$ n expands before A

from 2 $f(A) < f(B)$

$\therefore$ A expands before B

$\therefore$ n expands before B

- All ancestors of A expand before B
- A expands befor B
- A* search is optimal Solution

  only if $0 \leq h(n) \leq g(n)$

* Uniform – Cost expands equally in all directions.

* A* expands mainly toward the goal, but does hedge its bets to ensure optimality.

* A* Applications

- Pathing / routing problems
- Video games
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition.
- - - - -

* Creating Admissible Heuristics

- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics.

- Often, admissible heuristics are solutions to relaxed problems where new actions are available

- Inadmissible heuristics are often useful too.

<u>Example</u> : 8 Puzzle

7	2	4
5		6
8	3	1

Start state

3	7	1	
2	4	5	
	8		6

Action

	1	2
3	4	5
6	7	8

Goal State.

What are the states?
all possible cases that I can move any Number from its location to another one if it is next to the free space.

How many states?
$8! =$

What are the actions?
moving the number to free space next to it

How many successors from the start state
maximum 4 moves minimum 2 moves (successors)

What should the cost be?
1 for every move.

Heuristic: Number of tiles misplaced.
h(start) = 8

Why it is admissable?

[1] There is a relaxed-problem heuristic
(easiest solution, but need more work)

[2] Direct Move.

- move every number from start state to final state
  h = number of moves
  $h(1) = 3$ (moves to reach to its correct state at goal state)
  $h(2) = 1$, $h(3) = 2$ and so on

Total $h = 18$ for the start state of the example to reach final state

→ For the same goal and any start state we get the same Total h

→ Admissable only if $0 \leq h(n) \leq g(n)$
heuristic less or equal to cost.

- How about using the actual cost as a heuristic?
  $$f(n) = h(n) + g(n) = 2\,g(n) = 2\,h(n)$$
  if $g(n) = h(n)$
  → we go back to UCS by a double cost value
  trade off : complex work & neglect heuristic

- With A^* : a trade-off between quality of estimate and work per node.

As heuristics get closer to the true cost, you will expand fewer nodes usually do more work per node to compute the heuristic itself.

* Trivial Heuristics , Dominance.

. Trivial heuristics

. bottom of lattice is the zero heuristic

. Top of lattice is the exact heuristic

→ IF h = Zero (no heuristics)

We go back to Uniform_Cost_Search

→ Dominance : $h_a \leqslant h_c$

$\forall n : h_a(n) \geqslant h_c(n)$ For all nodes

we choose $h_a(n)$ (the best one)

→ Heuristics from a semi_lattice :

. Max of admissible heuristics is admissible

$$h(n) = \max (h_a(n) , h_b(n))$$

we choose the max of them, (the best sol)

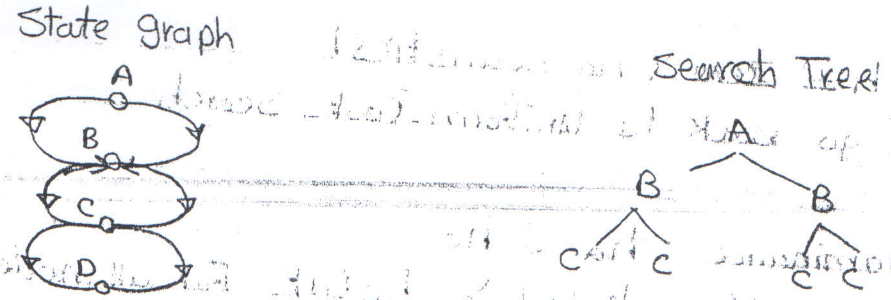
→ Heuristics equal to exact cost ,

we go back to Uniform Cost search

by double cost more work

* Graph Search.

→ Failure to detect repeated States Can Cause exponential more work



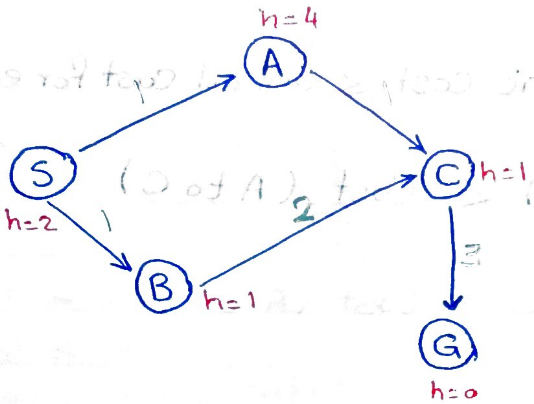
- Idea : never expand a state twice
- Implementation
  - Tree Search + Set of expanded States ('closed set')
    - Expand the Search tree node by node, but never expand a node twice.
    - Before expanding a node, check to make sure its state has never been expanded before.
  - If not now, skip it, If new add to closed set
- Important
  - Store the closed set as a set, not a list
  - Complete :

    Yes, If goal exists, we will find it
  - Optimality :

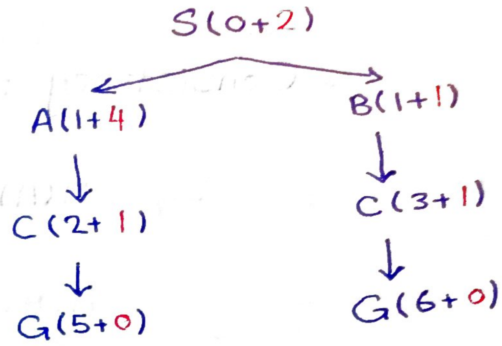
    No,

A^* Graph Search Gone Wrong?

→ Graph :⇒ Never expand a state twice
A^* :⇒ Combining UCS and Greedy Search



State Space graph



Search Tree.

→ At First S is expanded as $f = (0+2) = 2$
→ S children ⇒ A as $f = 1+4 = 5$
⇒ B as $f = 1+1 = 2$ So B is expanded
→ B children ⇒ C as $f = 3+1 = 4 < f_A = 5$ So C is expanded
→ C children ⇒ G as $f = 6+0 = 6 > f_A = 5$ So go to expand A
→ A children ⇒ C : it is expanded before so we can't expand it again ⇒ go to expand G as $f = 6$

★ here, we take the wrong way as f is the biggest
this problem caused as we can't expand one state twice.

★ To Solve This Problem, The Algorithm must be Admissible and Consistent. .

★ Main Idea: Estimated heuristic Costs $\leq$ actual Costs.

- Admissibility : heuristics Cost $\leq$ actual cost to goal

$$h(A) \leq \text{actual Cost from } A \text{ to } G$$

↳ Backword Cost

- Consistency : heuristic cost $\leq$ actual cost for each arc

$$h(A) - h(C) \leq \text{Cost } (A \text{ to } C)$$

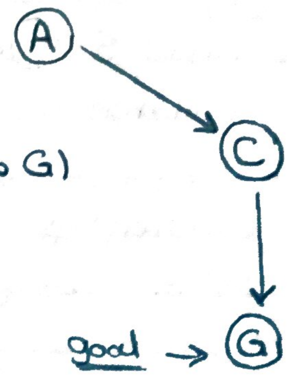
$$h(A) \leq h(C) + \text{Cost } (A \text{ to } C)$$

↳ Forward Cost.

→ Proof that IF The Algorithm is Consistent, it is Admissible.

- $\text{Cost } (A \text{ to } C) \leq \text{Cost } (A \text{ to } G)$ → ①

- $h(C) \leq \text{Cost } (C \text{ to } G) \leq \text{Cost } (A \text{ to } G)$ → ②



$\because$ The Algorithm is Consistent

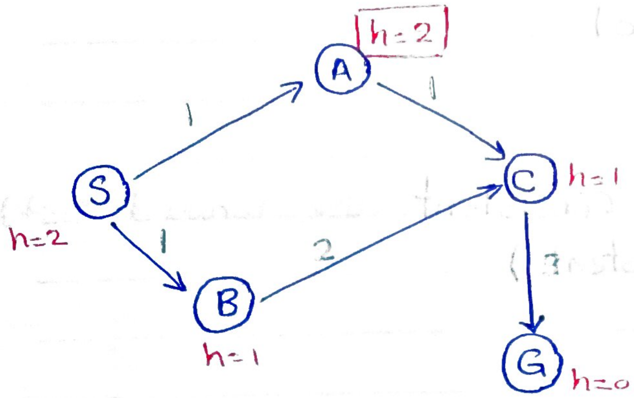
$\therefore$ $h(A) \leq h(C) + \text{Cost}(A \text{ to } C)$
From ① and ②

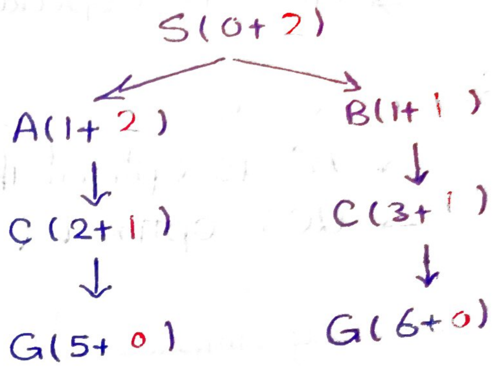
$\therefore$ $h(A) \leq \text{Cost}(A \text{ to } C) + \text{Cost } (C \text{ to } G)$

$\because$ $\text{Cost } (A \text{ to } C) + \text{Cost } (C \text{ to } G) = \text{Cost } (A \text{ to } G)$

$\therefore$ $h(A) \leq \text{Cost } (A \text{ to } G) \Rightarrow$ $\therefore$ the Algorithm is Admissable.

Solving the problem of the Last example





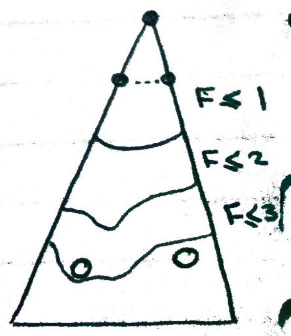
→ $h(A) \leq Cost(A \text{ to } G)$
$h(A) \leq 4$
→ $h(A) \leq h(C) + Cost(A \text{ to } C)$
$h(A) \leq 2$

∴ $\boxed{h(A) \text{ at max} = 2}$

→ expand S → expand B → expand A → expand C → expand G
→ optimal Solution.
we reach to goal with min f for each step.

- Consequences of Consistency:
  - → The f value along a path never decreases.
  - → A* graph Search is Optimal.

- Sketch : Consider that A* does with a consistent heuristic:
  - Fact 1 : In tree search, A* expands nodes in increasing total f value (f-contours)



  - Fact 2 : For every State S, nodes that reach S optimally are expanded before nodes that reach S suboptimally.

  - Result: A* graph Search is optimal

* Tree Search:
  - → A* is optimal if heuristic is admissible. (uses backword cost)
  - → UCS is a special case (h=0)

* Graph Search:
  - → A* is optimal if heuristic is Consistent. (uses Forward cost)
  - → UCS optimal (h=0 is Consistent)

* Consistency Implies Admissibility.

* In general, most natural admissible heuristics tend to be Consistent, especially if from relaxed problems.